VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Rekonstrukce 3D scény ze série snímků

Multi-View Stereo Reconstruction from Image Sequence

2014 Martin Kusyn

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání diplomové práce

Student: **Bc. Martin Kusyn**

Studijní program: N2647 Informační a komunikační technologie

Studijní obor: 2612T025 Informatika a výpočetní technika

Téma: Rekonstrukce 3D scény ze série snímků
Multi-View Stereo Reconstruction from Image Sequence

Zásady pro vypracování:

Cílem práce je vytvořit hustou síť rekonstruující objekty v záběru z mnoha různých pohledů kalibrované kamery. Výstupem by měla být otexturovaná polygonální síť vhodná pro další zobrazení.

1. Seznamte se s metodami z oblasti Multi-View Stereo Reconstruction.
2. Vybranou metodu implementujte v jazyce C++.
3. Na vhodném příkladě proveďte rekonstrukci nasnímané scény.
4. Interaktivně zobrazte vygenerovaný model (ideálně pomocí OpenGL).

Seznam doporučené odborné literatury:

[1] Michael Goesele, Steven M. Seitz and Brian Curless. Multi-View Stereo Revisited. In Proceedings of CVPR 2006, New York, NY, USA, June 2006.
[2] Richard Szeliski. Computer Vision: Algorithms and Applications. Springer. September 2010.
[3] Steven M. Seitz, Brian Curless, James Diebel, Daniel Scharstein, Richard Szeliski. A Comparison and Evaluation of Multi-View Stereo Reconstruction Algorithms, In Proceedings of CVPR 2006, New York, NY, USA, June 2006.
[4] Michael Goesele, Noah Snavely, Brian Curless, Hugues Hoppe, Steven M. Seitz. Multi-View Stereo for Community Photo Collections. In Proceedings of ICCV 2007, Rio de Janeiro, Brasil, October 2007.
[5] Furukawa, Y., Curless, B., Seitz, S.M., Szeliski, R. Towards Internet-scale multi-view stereo. In Proceedings of CVPR 2010, San Francisco, CA, USA, June 2010.

Formální náležitosti a rozsah diplomové práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí diplomové práce: **Ing. Tomáš Fabián**

Datum zadání: 16.11.2012

Datum odevzdání: 07.05.2014

doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 6. května 2014 .....

Rád bych na tomto místě poděkoval kolegům z firmy Argutec, s.r.o. za jejich psychickou podporu ve vypjatém období před dokončením této práce.

Abstrakt

Tato diplomová práce se zabývá prostorovou rekonstrukcí ze série kalibrovaných snímků založenou na principech Structure from Motion. První část práce představuje základní principy Structure from Motion a několik algoritmů a postupů řešících jednotlivé kroky celkového postupu rekonstrukce. V druhé části práce je navržena demonstrační aplikace. Samotná implementace navržené aplikace je uskutečněna za použití knihoven OpenCV a PCL. Na závěr jsou výsledky generované navrženou aplikací porovnány s výsledky v současnosti nejpokročilejšího uživatelsky přístupného řešení Autodesk 123D Catch.

Klíčová slova: 3D rekonstrukce, Structure from Motion, OpenCV

Abstract

This thesis focus on the spatial reconstruction from the calibrated image set based on the principles of the Structure from Motion pattern. First part of the thesis introduces the basic principles of Structure from Motion pattern and several common algorithms used for solving the particular steps of the overall workflow. In the second part of the thesis the demonstrational application is proposed and discussed. The implementation of the proposed application is done with the usage of OpenCV and PCL libraries. Finally, the results of the demonstrational application are compared with the current state of the art consumer-level solution Autodesk 123D Catch.

Keywords: 3D reconstruction, Structure from Motion, OpenCV

Seznam použitých zkratek a symbolů

CCD	–	Charge-Coupled Device
DOF	–	Degrees Of Freedom
DoG	–	Difference of Gaussians
DTL	–	Direct Linear Transformation
FAST	–	Features from Accelerated Segment Test
FLANN	–	Fast Library for Approximate Nearest Neighbors
LoG	–	Laplacian of Gaussian
ORB	–	Oriented BRIEF and Rotated BRIEF
RANSAC	–	Random Sample Consensus
RMS	–	Root-Mean-Square
SfM	–	Structure from Motion
SIFT	–	Scale-Invariant Feature Transform
SOCP	–	Second-Order Cone Programming
SURF	–	Speeded-Up Robust Features
SVD	–	Singular Value Decomposition

Contents

List of Tables

List of Figures

1 Introduction

The problem of automated spatial reconstruction of the real-life objects is one of the biggest challenges in the field of computer vision since the computers reached the computing power levels high enough to visualize those spatial models. The evolution of spatial reconstruction algorithms is slow and highly dependent on available technologies.

The process of digitizing the objects using the professional tools already reached fine results with high precision, however the technologies used are still very expansive and inaccessible. The solutions that are using the current mainstream still do not achieve the satisfactory precision.

However, with increasing camera sensor resolution and advances in the lens manufacturing technology, the possibility of using standard mainstream cameras for performing spatial reconstruction becomes more realistic.

This thesis focuses on the possibilities of spatial reconstruction using the common cameras available on the market, without using any supplementary devices or constructions. The goal of this thesis is to examine the currently existing solutions, compare their possibilities, suggest own solution and confront it with existing applications.

Whole process of spatial reconstruction consists of several separable steps that can be solved using several existing algorithms. Chapter 2 describes the theory behind the algorithms that can be used in order to solve the individual steps.

The proposed solution is discussed and compared with the model solution in the chapter 3, whilst the implementation details are particularized in chapter 4. Chapter 6 focuses on evaluation of the proposed solution's results and their comparisons with the results of the already existing applications.

1.1 Spatial reconstruction methods

The problem of acquiring three-dimensional structure of existing objects is possible to solve in several different ways based on several different principles. Basically the methods are divided into active and passive methods.

1.1.1 Active methods

Active methods actively interfere with reconstructed object. This can be done either mechanically or radiometrically. As an example of mechanical method can be used a contact scanner. This method is based on the contact with the surface of an object determined for scanning.

Radiometrical methods uses are based on a principle of emiting radiance towards the scanned object and measuring the reflected parts. To this category belong methods working with ultrasound, microwaves, laser measurment or methods based on optical effects like refraction of modulated light. Radiometrical methods are usually very precise, since the measurement is performed in controlled conditions and all measurement parameters are known.

1.1.2 Passive methods

Passive methods rely on detection of reflected ambient radiation, mostly the visible light. Passive detection methods are cheaper and easier to perform since they mostly require common hardware. In controlled conditions, passive methods are able to deliver precise results as well. Passive methods are also divided into three categories.

Stereoscopic systems calculates the differences in images obtained using two cameras positioned slightly apart. The estimation of spatial positions of mutual points in images are based on knowledge of intrinsic cameras parameters. These methods are based on principle similar to recovering spatial data by human eyes.

Photometric methods usually use single camera to obtain multiple images of object in various lighting conditions and estimate surface normals of object for each pixel.

Silhouette techniques work with set of images taken around the object against the contrasted background. The silhouettes of object are outlined from each image and connected together using the knowledge of camera positions. These approaches are quite precise, but cannot detect concavities of scanned objects.

1.2 Structure from motion

Structure from motion (SfM) belongs to the category of passive stereoscopic systems. The spatial reconstruction is based on relative camera pose estimation of each image pair from the input image set, followed by estimation of the absolute camera poses and triangulation.

This is the main difference between the SfM and multi-view spatial reconstruction using calibrated rigs, where the absolute camera positions are exactly known. Due to this fact, the spatial reconstruction using the calibrated rigs produces much more precise results.

1.3 State of the art solutions

The recent advances in the computing power allowed for the development of applications that are able to perform the whole processing of the spatial reconstruction. The two selected applications described in following paragraphs represents the complete solutions of spatial reconstruction.

BUNDLER is the academic solution with broad range of settings while the Autodesk's 123D Catch represents the commercially available customer application with simplified user-friendly interface with minimal set of parameters.

1.3.1 BUNDLER

BUNDLER represent the most common approach of multi-view structure from motion solutions based on sequential adding the cameras to the scene and increasing the accuracy of 3D point position estimation. This approach has several weaknesses. As was pointed out in [9], the quality of scene reconstruction heavily depends on initial image

pair choice and results are strongly affected by selection of used images and their order. The result structure also often suffer from drift (error build-up)

Through all the weaknesses, this methods are also able to provide decent results. The system released in 2008 as Bundler - one of currently most reliable solutions using this approach - was presented in [24] focused on large-scale reconstructions.

1.3.2 123D Catch

123D Catch is one of the first commercially available product oriented to simple spatial reconstruction. It is provided in a form of an internet service provided by Autodesk as a part of 123D modeling suite.

It allows users to reconstruct the 3D model simply by uploading the pictures via web interface. The source pictures have to fulfil the basic request only – for optimal results the input set should contain loop of the pictures in small increments around the object and the second loop from different angle, if possible. The object have to be static and it should be placed on a contrast and ragged surface, when the pictures are acquired.

Pictures are processed on the server-side and the final 3D model is then accessible to user for further modifications, it is also possible to prepare the model for 3D printing and order it directly. The process is complete and includes all the steps of reconstruction including the texturing of the final model.

2 Basic principles

The solution for 3D scene reconstruction from image sequence is mostly based on structure from motion principles. The usual input for methods solving this problem is the set of the images and optionally the camera calibration data.

The basic pipeline consists of several steps.

1. Finding the corresponding keypoints in the image pairs.
2. Estimation of the relative positions and rotations between camera pairs according to found corresponding keypoints.
3. Point position reconstruction in 3D scene using the estimated camera positions and rotations.
4. Surface triangulation.
5. Texturing.

Every step of the pipeline can be divided to several substeps, each solving one of the parts of the problem and there is already several solutions solving each step of the pipeline.

First step of the pipeline can be divided to two basic steps - features extraction and correspondence matching. OpenCV library implements several methods for both steps. The implemented algorithms are robust and quite reliable.

In the second step of the pipeline, there are some essential differences based on provided input. For sequence of uncalibrated pictures it is necessary to find the relative calibration. In comparison to the solution for the calibrated image sequence, the solution for the uncalibrated image sequence requires more corresponding points between the images to estimate the positions, focal length and other parameters of cameras. Once the relative positions amongst the image pairs are estimated, it is necessary to filter out the false detected image pairs and estimate the absolute camera positions and rotations.

Third step of the pipeline consists of estimating the positions of found the corresponding points in the images in the 3D scene and creating the point cloud structure for further processing. Fourth step – surface triangulation – is also quite reliably covered by existing solutions. The Point Cloud Library provides several reliable and robust algorithms that can be used to create the final mesh from provided point cloud acquired in previous step.

Once the surface is triangulated, it is desirable to cover the newly formed mesh with texture. This texture can be obtained using several approaches. The simplest solution is to assign a color to each point and interpolate the color of connected points. This method is really simple, but for a point clouds with high density it can provide a decent result.

More advanced solutions are based on UV mapping of the texture from the input images on the polygons followed by interpolation of the textures mapped on the same polygon. From the description above, it is obvious that the most problematic point in the whole process is the estimation of relative camera positions and rotations. There are several existing solutions for this problem, however there is a lot of space for improvement. Therefore this work will focus mostly on this step.

2.1 Camera calibration

In order to extract the spatial information from images, camera calibration is a necessary step. The images obtained with real cameras often suffer from several deficiencies that aggravates the process of spatial reconstruction.

The calculations in scene reconstruction are usually based on simple pinhole camera model that does not suffer from radial and tangential distortion, since the simplified pinhole camera model does not contain any lenses and all the calculations expect that all the light beams pass through the single point, when the image is acquired. This fact means that all the straight line in the real scene are straight in image as well.

Unlike the simplified pinhole camera model, the real cameras contains the set of lenses that are causing several types of distortion. Radial distortion causes that the straight lines in the pictures are bended, if they does not pass through the image center. That is caused by the spherical shape of lenses.

Tangential distortion is caused by the manufacturing defects. When the CCD in the camera is not exactly parallel with the lens, the lines from the real scene parallel to the image plane are not displayed as parallel in the image. Real cameras also suffer from other distortions e.g. chromatic aberration, vignette or blurring, however, in modern cameras these distortions are usually suppressed to large extent and aren't affecting the image quality so much. More details on image distortion caused by usage of lenses can be found in [8, 23].

2.1.1 Camera calibration basic principles and equations

The basic camera calibration problems and principles were described and discussed in the papers [25] and [26] by Z. Zhang.

The relationship between spatial point $\mathbf{X}$ and the image point x is following

$$\mathbf{s}x = \mathbf{A}[\mathbf{R}|\mathbf{t}]\mathbf{X}\,, \tag{1}$$

where $\mathbf{A}$ is camera intrinsic matrix, $\mathbf{R}$ is camera absolute rotational matrix and $\mathbf{t}$ is position vector. $\mathbf{R}$ and $\mathbf{t}$ forms the camera matrix P. The camera intrinsic matrix $\mathbf{A}$ is given by

$$\mathbf{A} = \begin{bmatrix} \alpha & \gamma & u_0 \\ 0 & \beta & v_0 \\ 0 & 0 & 1 \end{bmatrix},$$

where u_0 and $v_0$are the coordinates of principal point, α and β are the focal lengths for image's x and y axis and γ is the parameter describing the skewness of the image axis.

The equation (1) can be also expressed using homography matrix $\mathbf{H}$ as

$$\mathbf{s}x = \mathbf{H}\mathbf{X}\,,$$

where

$$\mathbf{H} = \mathbf{A}[\ \mathbf{r_1}\quad \mathbf{r_2}\quad \mathbf{t}\ ]\,. \tag{2}$$

$r_1$and r_2 denotes the first and second column of the rotation matrix R. Since the scale information was disposed, it is clear that $\mathbf{H}$ is defined up to scale factor. Denoting the homography matrix as $\mathbf{H} = [\ \mathbf{h_1} \quad \mathbf{h_2} \quad \mathbf{h_3}\ ]$, we can express the relationship (2) as

$$[\ \mathbf{h_1} \quad \mathbf{h_2} \quad \mathbf{h_3}\ ] = \lambda \mathbf{A} [\ \mathbf{r_1} \quad \mathbf{r_2} \quad \mathbf{t}\ ],$$

where $\lambda$is an arbitrary scalar. Knowing the r_1 and r_2 are orthonormal the following two constraints are given

$$\mathbf{h_1^T A^{-T} A^{-1} h_2} = 0\,. \tag{3}$$

$$\mathbf{h_1^T A^{-T} A^{-1} h_1} = \mathbf{h_2^T A^{-T} A^{-1} h_2}\,. \tag{4}$$

Homography has 8 degrees of freedom and there 6 intrinsic parameters (3 for both rotation and translation), but only two constraints on intrinsic parameters.

2.1.2 Camera calibration algorithm by Z. Zhang

The solution for camera calibration by Z. Zhang was released in 1998 in [25]. The Zhang's solution is based on observing a planar pattern from several different orientations – the minimum is two. The images can be obtained by moving the camera or the pattern and the motion need not to be known. The procedure consist of a closed-form solution of intrinsic and extrinsic parameters followed by non-linear refinement based on the maximum likelihood criterion. Along with the intrinsic parameters, the radial distortion coefficients are estimated.

The technique of camera calibration using Zhang algorithm is following:

- printing the calibration pattern and attaching to it planar surface,
- acquiring the images of model plane under different angles,
- feature point detection in the images,
- extraction of the five intrinsic and all extrinsic parameters using the closed-form solution,
- estimating the distortion coefficients,
- refining all parameters.

2.1.3 Intrinsic camera parameters estimation

Zhang starts with the camera intrinsic parameters using the closed-form solution. For this purpose the matrix $\mathbf{B}$ is defined as

$$\mathbf{B} = (\mathbf{A^T})^{-1} \mathbf{A^{-1}}\,.$$

Matrix $\mathbf{B}$ is symmetric, defined by 6D vector

$$\mathbf{b} = [\ B_{11} \quad B_{12} \quad B_{22} \quad B_{13} \quad B_{23} \quad B_{33}\ ]^{\mathbf{T}}\,.$$

Denoting the i^{th} column of homography matrix $\mathbf{H}$ as $\mathbf{h_i} = [\ \mathbf{h_{i1}} \quad \mathbf{h_{i2}} \quad \mathbf{h_{i3}}\ ]$, the following equation can be expressed

$$\mathbf{h_i^T B h_j} = \mathbf{v_{ij}^T b}\,.$$

The fundamental constraints (3) and (4) from given homography can then be rewritten as 2 homogenous equations in $\mathbf{b}$

$$\begin{bmatrix} \mathbf{v_{12}^T} \\ (\mathbf{v_{11}} - \mathbf{v_{22}})^{\mathbf{T}} \end{bmatrix} \mathbf{b} = 0\,.$$

Stacking above presented equations derived from the images of observed model plane, the matrix $\mathbf{V}$ can be formed by fulfilling the following constraint

$$\mathbf{Vb} = 0$$

In order to estimate the camera intrinsic parameters, it is enough to form the $\mathbf{V}$ matrix from one image, however for estimatimation of the skewness coefficients and to get unique solution for b, it is necessary to use more than three pictures.

The solution for $\mathbf{Vb} = 0$ is then equal to the eigenvector of $\mathbf{V^T V}$ associated with the smallest eigenvalue. Once $\mathbf{B}$ is estimated the extrinsic camera parameters$\mathbf{P} = [\mathbf{R}|\mathbf{t}]$ for each image can be computed as

$$\mathbf{r_1} = \lambda \mathbf{A^{-1} h_1}\,.$$

$$\mathbf{r_2} = \lambda \mathbf{A^{-1} h_2}\,.$$

$$\mathbf{r_3} = \mathbf{r_1} \times \mathbf{r_2}\,.$$

$$\mathbf{t} = \lambda \mathbf{A^{-1} h_3}\,.$$

To enhance the accuracy, the solution is obtained through the minimization of algebraic distance. For n given images of calibration pattern plane, there are $n \cdot m$ calibration points in total. It is assumed that calibration points are corrupted by noise, so the maximum likelihood is estimated by minimizing following functional

$$\sum_{i=1}^{n} \sum_{j=1}^{m} \|m_{ij} - \hat{m}(\mathbf{A}, \mathbf{R_i}, \mathbf{t_i}, \mathbf{M_j})\|^2\ , \tag{5}$$

where $\hat{m}(\mathbf{A}, \mathbf{R_i}, \mathbf{t_i}, \mathbf{M_j})$ is the projection of the point $\mathbf{M_j}$ in image i. The rotational matrix and translation vector correspondingto the image i are represented as $\mathbf{R_i}$ and $\mathbf{t_i}$. The minimization of above mentioned functional is a nonlinear minimization problem solved by Levenberg-Marquardt algorithm presented in [13].

2.1.4 Radial distortion

The most significant lens distortion influencing the images captured using the real life cameras is the radial distortion caused by the curved surfaces of lenses used in the camera objectives. Radial distortion is not the only distortion contained in the final images, but it is the most significant one. For this reason it is usually assumed that the distortion function is dominated by the radial components.

Assuming that (u, v) are the ideal (distortion-free) pixel image coordinates and $(\breve{u}, \breve{v})$ are the corresponding real image coordinates, the ideal points are the projections of the model point through the pinhole camera model. The same notation is applied on the ideal (distortion-free) coordinates (x, y) and distorted normalized image coordinates $(\breve{x}, \breve{y})$.

$$\breve{x} = x + x[k_1(x^2 + y^2) + k_2(x^2 + y^2)^2]\,,$$

$$\breve{y} = y + y[k_1(x^2 + y^2) + k_2(x^2 + y^2)^2]\,,$$

where k_1 and k_2 are the radial distortion coefficients and the center of radial distortion is the same as the principal point.

Assuming that $\gamma = 0$ and the relationships

$$\breve{u} = u_0 + \alpha\breve{x} + \gamma\breve{y}\,,$$

$$\breve{v} = v_0 + \beta\breve{y}\,,$$

the following equations can be expressed

$$\breve{u} = u + (u - u_0)[k_1(x^2 + y^2) + k_2(x^2 + y^2)^2]\,. \tag{6}$$

$$\breve{v} = u + (v - v_0)[k_1(x^2 + y^2) + k_2(x^2 + y^2)^2]\,. \tag{7}$$

Estimating radial distrortion by alteration

One strategy to estimate k_1 and k_2 after already having estimated the other parameters that helps to estimate the ideal pixel coordinates (u, v). Substituting the coordinates into equations (6) and (7) the following equation set is obtained

$$\begin{bmatrix} (u - u_0)(x^2 + y^2) & (u - u_0)(x^2 + y^2)^2 \\ (v - v_0)(x^2 + y^2) & (v - v_0)(x^2 + y^2)^2 \end{bmatrix} \begin{bmatrix} k_1 \\ k_2 \end{bmatrix} = \begin{bmatrix} \breve{u} - u \\ \breve{v} - v \end{bmatrix}.$$

Stacking the equation for m points in n images, the $2mn$ equations are obtained and can be expressed in matrix form as $\mathbf{D}k = k$, where $k = [k_1, k_2]^{\mathbf{T}}$. The linear least-square solution is then given by

$$k = (\mathbf{D}^{\mathbf{T}}\mathbf{D})^{-1}\mathbf{D}^{\mathbf{T}}d\,.$$

Once k_1 and k_2 are estimated, the parameters estimation can be refined by solving (5) with $\hat{m}(\mathbf{A}, \mathbf{R_i}, \mathbf{t_i}, \mathbf{M_j})$ replaced by (6) and (7). This procedures can be alternated until convergence.

2.2 Feature extraction

In order to retrieve mutual points, known as features, in a pair of images it is necessary to identify and pair interesting points in both images. The interesting points are objects in the image that are easy to identify (e.g. corners or edges) and usually lie in high-contrast areas. The relative position of features in both pictures should not change. Currently most used and robust methods for feature extraction are SIFT, SURF and FAST.

2.2.1 Scale-invariant feature transform (SIFT)

SIFT is a feature extraction algorithm presented by David G. Lowe in [14]. Unlike the preceding algorithms (e.g. Harris corner detector) SIFT algorithm is not only rotation-invariant, but also scale-invariant. To solve this, SIFT algorithm extracts keypoints and computes its descriptors. The simplified description of SIFT algorithm can be found in [2].

Detection of extremes in scale-space

In order to retrieve candidate features, SIFT uses Difference of Gaussians (DoG), which approximates more complicated Laplacian of Gaussian (LoG) calculation. To obtain DoG, the Gaussian pyramid is prepared. Image is scaled in several octaves and each octave is blurred into several images using Gaussian blur with different strengths.

Once Gaussian pyramid is prepared, the neighbor image couples in octaves are differed. These differed images represents DoG. DoG images are searched for local extremes and compared to the other scales. If local extrema on the position is found in other scales, the point is considered as a potential keypoint.

Keypoint filtering

Potential keypoints are filtered to eliminate points lying in edges and low contrast areas. Points in low contrast areas are filtered using Taylor series expansion of scale space for retrieving more accurate location of extrema and checks if its intensity is higher than threshold.

Since the DoG has higher responses for edges, the 2x2 Hessian matrix is used to compute the principal curvature and compare eigenvalues ratio to given threshold. Remaining points are considered as strong interest points.

Orientation assignment

Each point is assigned an orientation in order to achieve rotational invariance, based on the point's circular neighborhood histogram. The orientation vector is determined using

the highest peak value and any peaks over 80% of its value. Orientation assignments contributes to the stability of features matching.

Keypoint descriptors

Each keypoint is assigned 128 dimension vector describing its neighborhood relatively to the previously assigned orientation. This ensure the rotational invariance.

Descriptor itself describes the orientation histograms in keypoint's 16x16 neighborhood divided into 16 sub-blocks. For each sub-block, 8 dimensional orientation histogram is created – 128 values available in total.

Keypoint matching

Finally, keypoints are matched by comparing the most similar descriptors computing the Euclidean distance. In some cases, second closest match can be very near to the first. In this case, it is not possible ambiguously determine the correct match. In this case, match is rejected if the ratio of closest match to second closest match is greater than 0.8.

2.2.2 Speeded up robust features (SURF)

SURF algorithm presented in [7] is sped up version of SIFT. In comparison performed in [5], SURF is 3 times faster than SIFT, but keeps similar quality of detection. SURF handles well images with blurring and rotations, but has problems with illumination and viewpoint changes.

According to [3], the SURF algorithm uses Box Filter to approximate the LoG. Advantage of this technique is that the convolution with box filters can be easily computed using the integral images. Instead of using Gaussian pyramid with different resolutions, the stack of same resolution images is created. Hessian matrix determinant is used for detection of interesting points.

Wavelet responses in horizontal and vertical directions are used for orientation assignment in SUFR algorithm. The dominant orientation is determined by computing the sum of all responses in 60° sliding orientation window in the graph of computed Gaussian weights of wavelet response.

In many occasions it is not necessary to compute the orientation. In situations when object on compared images does not rotate (e.g. panoramas), the features are always oriented in the same direction. This fact led to the simplification of SURF algorithm called Upright-SURF (also referred as U-SURF).

In this case, all the features are set to the same orientation. Skipping the orientation estimation improves the speed of the process while keeping the algorithm robust for detection of features with orientation difference lower than 15°.

The principle of keypoint descriptor calculation is similar to SIFT, but wavelet responses are used instead of orientation histograms. Neighborhood of 20x20 around the keypoint is divided into 16 subregions. For each subregion horizontal and vertical

wavelet responses are computed and a 4-vector describing the subregion is created. This gives the SURF feature descriptor with 64 dimensions in total.

Another performance tweak is provided by using the sign of Laplacian already obtained during the Hessian matrix computation. Sign of Laplacian distinguishes whether the keypoint is darker or lighter than the surrounding. Knowing the fact the matching keypoints have same sign of Laplacian, the keypoint comparing phase is noticeably sped up.

2.2.3 Features from accelerated segment test (FAST)

FAST algorithm is a corner detection method optimized for computational efficiency and is suitable for real-time applications. Even though FAST algorithm is optimized for time efficiency, according to the [5], the quality of detection is very good – lots of features are detected while the average tracking error is acceptable. FAST was presented in 2006 by E. Rosten and T. Drummond in [22].

FAST detector uses relatively simple method to find the interesting points. The pixel is selected and circle of 16 pixels around it is tested. Selected pixel is determined as corner if the circle around it contains continuous sequence of 12 pixels that are brighter or darker than intensity of inspected pixel and predetermined threshold value.

The simplified description of FAST algorithm and its usage can be found in [1].

High-speed test

To speed up the process, the high-speed test is executed. High-speed test uses only 4 pixels of the circle around observed point – pixels a position 1, 9, 5 and 13. First pixels at 1 and 9 are tested if they fulfill the condition brightness difference with same result. If the condition is valid, pixels 5 and 13 are checked as well. If the observed pixel is corner, at least three of these points must fulfill the brightness condition with same result.

High-speed test simplifies the calculation, but suffers from several weaknesses. First of all, it is not ensured that the condition of 12 continuous pixels in circle around observed points fulfill the condition, so the standard test must be run as well. Either the choice of pixels is not optimal and effective in order to detect most of the corners in the image. Also the multiple features are detected in close areas.

2.3 Feature matching

Once the features are extracted, it is necessary to find the corresponding pair of features in the images. The simplest way to perform feature matching is to compare each pair of feature descriptors and match the closes ones. This approach does provides good results, however it is highly ineffective.

2.3.1 Standard types of feature matchers

More advanced methods usually use structures for sorting the descriptors. The paper [18] by M. Muja and D. Lowe describes the usages of the randomized kd-trees and hier-

archical k-mean trees. The performance improvement is noticeable mostly on large sets of features.

Even through good results and decent processing speeds, these methods still does not consider the relative positions of the points in scene – this is suitable in the case of searching the images for some object, however in the case of images with repeating patterns many false matches can be detected.

2.3.2 Filtering the feature matches

In order to eliminate the false matches it is necessary to use filtering. Simplest solutions compare the distance of the matched points in two images. Good results can be achieved restricting the minimal and maximal distance between matched image points.

In the more complex situations the minimal and maximal distance restriction can be insufficient, since the limits are static for whole image pair areas. If the case that distances between matches are large in one part of view and small in another, this method of filtering will not provide the optimal output – significant amount of the matches will be filtered, or many false matches will remain unfiltered.

2.3.3 Feature matching using the Optical Flow

Optical Flow pattern is widely used in the field of computer vision. Its main objective is to detect the movements in the sequence of images. The knowledge of movement then can be used for video stabilization, video compression, motion tracking, heat mapping, etc.

According to [4] the optical flow pattern works on following assumptions:

- The pixel intensities of an object do not change between consecutive frames.
- Neighboring pixels have similar motion.

The movement of the point $\mathbf{I}(x, y, t)$ where t denotes time can be expressed as

$$\mathbf{I}(x, y, t) = (x + dx,\, y + dy,\, t + dt)\,, \tag{8}$$

where dx and dy means the distance difference and dt denotes the time difference between two consequent frames.

Applying the taylor series approximation on right side of the equation (8), removing the common terms and dividing by dt the Optical flow equation is obtained:

$$f_x u + f_y v + f_t = 0\,, \tag{9}$$

where

$$f_x = \frac{\partial f}{\partial x}\,,$$

$$f_y = \frac{\partial f}{\partial y}\,,$$

$$u = \frac{dx}{dt}\,,$$

$$v = \frac{dy}{dt}\,.$$

The f_x an f_y are image gradients and f_t is the gradient along time. u and v are the unknowns, that can be solved using several methods – e.g. Lucas-Kanade.

Lucas-Kanade method

As was stated before, the optical flow pattern is based on assumption that Neighboring pixels have similar motion. The method presented in [15] by D. Lucas and T. Kanade uses 3x3 patch around the point and assumes that all 9 points have same motion. Having $(f_x,\, f_y\,, f_t)$ for the 9 points gives 9 equations (9) with two unknowns. The final solution is obtained using the least square fit method

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} \sum_i f_{x_i}^2 & \sum_i f_{x_i} f_{y_i} \\ \sum_i f_{x_i} f_{y_i} & \sum_i f_{y_i}^2 \end{bmatrix}^{-1} \begin{bmatrix} -\sum_i f_{x_i} f_{t_i} \\ -\sum_i f_{y_i} f_{t_i} \end{bmatrix}.$$

Sparse Optical Flow and feature matching

Lucas-Kanade method for sparse feature set – only the most distinct features are used for the optical flow calculation. These features are used to calculate the optical flow vectors of movement. Once the vectors of movement are obtained, it is possible use them for filtering the other matched features. If the approximate vector of movement is known, the points that are too far from the expected positions are filtered.

2.4 Relative pose calculation

The relative pose calculation is estimated using the relationships known from the epipolar geometry. In order to estimate the relative pose it is necessary to perform several steps. In case that the images were taken with constant intrinsic parameters it is possible to estimate the essential matrix using the knowledge of the previously computed calibration matrix.

2.4.1 Epipolar geometry

In [11] by Hartley and Zisserman the epipolar geometry between two views is described as the geometry of intersection of image planes with pencil of planes having baseline as the axis.

To explain basics of epipolar geometry, the knowledge of several basic terms is crucial. Each camera of image pair is defined by its **center** and **image plane** containing the flattened projection of real world. The line connecting the camera centers is known as **baseline**.

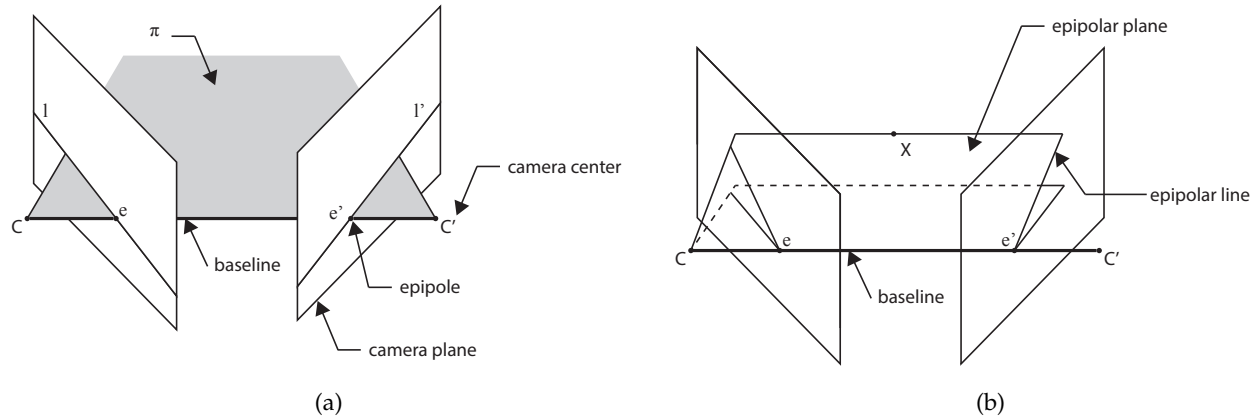


Figure 1: Epipolar geometry basics

The point of intersection of the baseline with the image plane is called **epipole**. Epipole is also the image of other's camera center in image plane, and it is the vanishing point of the baseline direction.

Epipolar plane is any plane containing the baseline. There is infinite amount of epipolar planes rotated around the axis determined by baseline.

Epipolar line is then the intersection of the epipolar and image plane. All epipolar lines intersect at each of the epipoles of both image pair.

The figure 1 illustrates the basic relationships in epipolar geometry. If the epipolar plane containing the spatial point $\mathbf{X}$ and the baseline, the projections x and x' of $\mathbf{X}$ will lie on the epipolar lines e and e'.

In case of the spatial reconstruction, the camera centers, image planes and projected points x and x'. The condition of the epipolarity ensures that the lines leading from camera centers to the projected points will intersect at some point, since both lines lie in one epipolar plane. The point of the intersection is the sought spatial point.

2.4.2 The Fundamental and Essential matrix

The epipolar geometry is intrinsic projective geometry between two views independent of scene structure and depends only on camera's internal parameters and relative pose. Intrinsic geometry is encapsulated by fundamental matrix commonly known as F, that describes the relationship between point x and corresponding point x' in stereo image pair. For each point x the Fx describes the epipolar line in second image from the image pair on which the corresponding point x' must lie. This fact implies following equation.

$$\mathbf{x'^{T}Fx} = 0\,. \tag{10}$$

The specialization of fundamental matrix used for computation with normalized cameras is called essential matrix. The relationship between fundamental matrix and essential matrix is described by

$$\mathbf{E} = \mathbf{K'^{T}}\mathbf{F}\mathbf{K}\,, \tag{11}$$

where calibration matrices K and K' describe the internal parameters of image pair's first and second camera.

The essential matrix encapsulates the information about the calibrated camera matrices. Consider the pair of normalized cameras $\mathbf{P} = [\mathbf{I}|0]$ and $P' = [\mathbf{R}|\mathbf{t}]$ the essential matrix can be described as

$$\mathbf{E} = [\mathbf{t}]\mathbf{x}\mathbf{R}\,,$$

wheret and $\mathbf{R}$ stands for relative translation and relative rotation between the cameras $\mathbf{P}$ and $\mathbf{P'}$.

2.4.3 Five-point pose estimation

The relative pose estimation of two calibrated camera with constant intrinsic parameters can be determined from five known matched features. The goal of five-point estimation is to find the essential matrix, which contains the information about rotation and translation that can be extracted to corresponding motion vectors.

Five-point pose estimation works on several presumptions - for two calibrated cameras with constant intrinsic matrix $\mathbf{K}$ the two corresponding points $\mathbf{x}$ and $\mathbf{x'}$ are related by fundamental matrix $\mathbf{F}$ as was stated by equation (10).

Valid fundamental matrix $\mathbf{F}$ must satisfy the cubic singularity condition

$$det(\mathbf{F}) = 0\,.$$

For fully-calibrated cameras the fundamental matrix $\mathbf{F}$ is reduced to an essential matrix $\mathbf{E}$

$$\mathbf{K^{-T}}\mathbf{E}\mathbf{K^{-1}} = \mathbf{F}\,.$$

Unlike fundamental matrix with seven degrees of freedom (DOF), essential matrix has only five DOFs. Consequently, the essential matrix must satisfy two more constraints characterized by equation:

$$2\mathbf{E}\mathbf{E}^{\mathbf{T}}\mathbf{E}\text{–}trace(\mathbf{E}\mathbf{E}^{\mathbf{T}})\mathbf{E} = 0\,. \tag{12}$$

Using above mentioned equation gives the nine equations in elements of essential matrix $\mathbf{E}$. With five corresponding points, there are five epipolar equations (10) and two more equations from singularity condition (12), there is enough equations to estimate the essential matrix. More details on five-point pose estimation algorithm can be found in [20, 12].

2.4.4 Eight-point algorithm

Simplest method of fundamental matrix computation is the normalized eight-point algorithm originally presented by H. C. Longuet-Higgins in 1981 in [13]. The algorithm involves only construction and solution of set of the linear equations. Despite its simplicity the algorithm can perform extremely well.

The method works with 8 and more corresponded image pairs and it is assumed that the input data are normalized. The suggested normalization consists of translation and scaling of each image so that the centroid of the reference points is at the origin of the coordinates and RMS distance of the points from the origin is $\sqrt{2}$. Normalizing the input data improves the performance of the DLT algorithm used for homography estimation.

The normalized eight-point algorithm as described in [11] by Hartley and Zisserman also includes the method for enforcing the singularity constraint on fundamental matrix. The initially computed fundamental matrix $\mathbf{F}$ is replaced by the singular matrix $\mathbf{F}'$ that minimizes the difference $\|\mathbf{F}' - \mathbf{F}\|$. The process of the minimization is commonly done using the SVD, which is simple and rapid, however the method is not optimal numerically, since not all entries of $\mathbf{F}$ have the equal importance.

The eight-point algorithm proceeds as follows:

- normalization – transformation of the input data using transformation matrices $\mathbf{T}$ and $\mathbf{T}'$,
- finding the fundamental matrix $\mathbf{F}'$ for normalized input data,
  - linear solution – determining $\mathbf{F}$ from the set of linear equations,
  - constraint enforcement – replacing $\mathbf{F}$ with $\mathbf{F}'$ such that $det(\mathbf{F}') = 0$,
- denormalization – obtaining the fundamental $\mathbf{F}$ for original input data using transformation matrices $\mathbf{T}$ and $\mathbf{T}'$.

2.4.5 Five-point pose estimation algorithm by Nistér

The five-point pose estimation algorithm described by David Nistér in [20] becomes widely used in the field of computer vision and even though there were several attempts to enhance or simplify it, Nistér's solution can still be considered the standard solution for the five-pose estimation problem.

Nistér's five-point algorithm proceeds as follows:

1. The epipolar equations for the five corresponding points into 5x9 matrix.
2. Creating the 9x20 matrix corresponding to a monomial vector $[x^3, y^3, x^2y, xy^2, x^2z, x^2, y^2z, y^2, xyz, xy, xz^2, xz, x, yz^2, yz, y, z^3, z^2, z, 1]$.
3. Reducing the 9x20 matrix to an upper triangle form using Gauss-Jordan elimination.
4. Extraction of the determinants and assembling it into two 4x4 matrices.

5. Second stage elimination and obtaining the 10-th degree polynomial.
6. Solving the polynomial using the suitable method and obtaining 10 solutions for the z-variable.
7. Backwards substitution of the real roots.
8. Recovering the essential matrix and extracting corresponding motion vectors.

2.5 Essential matrix decomposition

According to [11], once the essential matrix $\mathbf{E}$ is estimated, it is possible to retrieve the camera matrices from it. Assuming that first camera matrix $\mathbf{P} = [\mathbf{I}|0]$, it is necessary to factor $\mathbf{E}$ into a product of skew-symmetric matrix $\mathbf{S}$ and rotation matrix $\mathbf{R}$. There are two possible factorizations of $\mathbf{E} = \mathbf{SR}$, since $\mathbf{R}$ can be expressed as (14) or (15).

$$\mathbf{S} = \mathbf{UZU^T}\,. \tag{13}$$

$$\mathbf{R} = \mathbf{UWV^T}\,. \tag{14}$$

$$\mathbf{R} = \mathbf{UW^TV^T}\,. \tag{15}$$

The above mentioned factorization determines the translation $\mathbf{t}$ as the part of camera matrix $\mathbf{P'}$. Up to scale from $\mathbf{S} = [\mathbf{t}]_x$. However, the Frobenius norm of (13) is $\sqrt{2}$. If $\mathbf{S} = [\mathbf{t}]_x$ including scale, than $\|\mathbf{t}\| = 1$. Since $\mathbf{St} = 0$ it follows that $\mathbf{t} = \mathbf{U}(0,0,1)^T = \mathbf{u_3}$, where $\mathbf{u_3}$ denotes the third column of $\mathbf{U}$.

The sign of $\mathbf{E}$ cannot be determined from the equation $\mathbf{E} = \mathbf{SR}$. Consequently also the sign of $\mathbf{t}$ cannot be determined. There are four possible choices of $\mathbf{R}$ and two possible signs for $\mathbf{t}$ corresponding to given essential matrix $\mathbf{E}$, thus there are four possible choices of $\mathbf{P'}$:

$$\mathbf{P'} = [\mathbf{UWV^T}|\mathbf{u_3}]\,.$$

$$\mathbf{P'} = [\mathbf{UWV^T}|-\mathbf{u_3}]\,.$$

$$\mathbf{P'} = [\mathbf{UW^TV^T}|\mathbf{u_3}]\,.$$

$$\mathbf{P'} = [\mathbf{UW^TV^T}|-\mathbf{u_3}]\,.$$

Cheirality check

In order to determine the camera matrix $\mathbf{P}'$ that corresponds to the true configuration the cheirality check described in [20] is performed. The cheirality constraint assumes that the scene points are lying in front of the cameras.

To check the cheirality constraint it is only necessary to triangulate one image point correspondence. The point $\mathbf{Q}$ is triangulated using the view pair $([\mathbf{I}|0], \mathbf{P}')$.

- If $c_1 = \mathbf{Q_3 Q_4} < 0$ the point lies behind the first camera.
- If $c_2 = (\mathbf{P}'\mathbf{Q})_3 \mathbf{Q_4} < 0$ the point lies behind the second camera.
- If $c_1 > 0$ and $c_2 > 0$, the selected $\mathbf{P}'$ corresponds to the true configuration.

2.6 Absolute pose calculation

Suppose the relative poses for all image couples are acquired, it is possible to calculate absolute camera positions. It is highly probable that there will be a great amount of spatially impossible image couples, especially in case of the circular set of images. It is impossible for a couple images that captures the object from opposite sides to form an image couple suitable for spatial reconstruction. These image couples are usually programmatically accepted while having a number of incorrectly matched features. For further processing, it is necessary to filter out the outlier image pairs to create consistent camera cycle.

Experiment performed as a part of [9] proves that in the case of decreasing distance between each pair of the cameras, the rotation estimation is significantly more stable than the translation estimation. With knowledge of this assumption it is reasonable to focus on filtering the outlier image pairs using the computed relative rotations as the metric, since the probability of its correct estimation is significantly higher.

There are two goals for absolute pose calculation to solve. First it is necessary to filter out the outlier image pairs, the second one is to find the most probable set of the camera transformations that reflect the real world setup.

Solving the outlier image pair issue is possible using the theory of graphs with suitably chosen quality metrics. For example, if the outlier filtering is based on the rotation estimations, it is possible to create a spanning tree of maximal axis rotations and then create cycles using the remaining rotations verifying if the newly formed cycle fulfils the predetermined condition – e.g. if the error in cycle's rotation reconstruction is lower than predetermined threshold. Usual approach to estimate the most plausible absolute camera rotations it to perform the minimization of the error amongst the remaining (inline) rotations.

2.7 Projection reconstruction

The elementary step of projection reconstruction is to estimate the spatial position of each matched feature of currently processed image pair. Same as the relative pose estimation,

the estimation of the point's spatial positions is based on the epipolar geometry principles described in section 2.4.1.

Hartley's and Sturm's paper [10] discuss the problems of projection reconstruction and presents several algorithms for spatial points triangulation. For the estimation of spatial position of point $\mathbf{X}$ from pair of corresponding image points $\mathbf{x}$ and $\mathbf{x}'$ the key relationships are the following two equations

$$\mathbf{x} = \mathbf{P}\mathbf{X}\,. \tag{16}$$

$$\mathbf{x}' = \mathbf{P}'\mathbf{X}\,. \tag{17}$$

Considering that the absolute camera matrices $\mathbf{P}$ and $\mathbf{P'}$ are already known after applying the five-point algorithm and absolute pose estimation, it is possible to triangulate the spatial position of the point $\mathbf{X}$. Triangulation itself can be processed using various possible algorithms, the most straightforward methods are linear triangulation methods. The above mentioned equations (16) and (17) gives three following equations for each of paired image points $\mathbf{x} = w(u, v, 1)^{\mathbf{T}}$:

$$wu = \mathbf{p^{1T}}\mathbf{X}\,,$$

$$wv = \mathbf{p^{2T}}\mathbf{X}\,,$$

$$w = \mathbf{p^{3T}}\mathbf{X}\,,$$

where w is the an unknown scale factor and $\mathbf{p^{iT}}$ denotes the i-th row of the $\mathbf{P}$.

Eliminating the scale factor w using the third equation, the following equations can be formed:

$$u\mathbf{p^{3T}}\mathbf{X} - \mathbf{p^{1T}}\mathbf{X} = 0\,.$$

$$v\mathbf{p^{3T}}\mathbf{X} - \mathbf{p^{2T}}\mathbf{X} = 0\,.$$

The relationships (16) and (17) can be also expressed in the form of $\mathbf{A}\mathbf{X} = 0$, where $\mathbf{A}$ is formed using the equations for points $x = w(x, y, 1)$ and $x' = w'(x', y', 1)$

$$\mathbf{A} = \begin{bmatrix} x\mathbf{p^{3T}} - \mathbf{p^{1T}} \\ y\mathbf{p^{3T}} - \mathbf{p^{2T}} \\ x'\mathbf{p'^{3T}} - \mathbf{p'^{1T}} \\ y\mathbf{p^{3T}} - \mathbf{p'^{2T}} \end{bmatrix}.$$

It's necessary to expect that the input image data will be noisy and the equations will not be satisfied precisely, therefore it is necessary to seek the best solution.

2.7.1 Linear-LS triangulation method

The Linear-LS method solves the problem by setting the $\mathbf{X} = (x, y, z, 1)^{\mathbf{T}}$. This reduces the set of homogenous equations $\mathbf{AX} = 0$ to a set of 4 non-homogenous equations with 3 unknowns that can be quite easily found using standard procedures, e.g. Singular Value Decomposition (SVD).

One of the presumption of Linear-LS method is that the X does not lie at infinity, otherwise it cannot be assumed that $\mathbf{X} = (x, y, z, 1)^{\mathbf{T}}$.

2.7.2 Linear-Eigen method

The Eigen method uses the unit eigenvector. This method focuses on finding $\mathbf{X}$ in order to minimize the $\|\mathbf{AX}\|$ subject to the condition $\|\mathbf{X}\| = 1$. The solution is the unit eigenvector corresponding to the smallest eigenvector of $\mathbf{A}^{\mathbf{T}}\mathbf{A}$. The problem can be solved using SVD.

2.8 Surface reconstruction

One of the possible approaches for further processing is to form obtained spatial points into point cloud – the structure of spatial point coordinates that can carry the additional information, e.g. the color of the point. In order to get a mesh out of the point cloud it is necessary triangulate the points. One of the methods for surface triangulation was presented in [16]. The algorithm is known as greedy surface algorithm.

The algorithm process the triangulation incrementally. For each point $\mathbf{P}$ the neighborhood is searched in order to find the point's nearest k-neighbors. The searched area is restricted by sphere with radius $r = \mu d_0$ where d_0 is distance to the closest neighbor of $\mathbf{P}$ and μ is user-specified constant. The neighborhood is then projected on a plane approximately tangential to the surface that is formed by the neighborhood. The points are then pruned by visibility and connected by edges to the P and surrounding points forming the triangles. The triangles must fulfill the maximum and optional minimum distance criterion.

3 Proposed solution

In the first part of this chapter the techniques and algorithms used in the model solution are presented and proposed solution differences are explained.

3.1 Model solution

As the model solution for this paper the work presented in Non-sequential structure from motion [9] was used. The practical contribution of above mentioned paper is improved handling of short baselines, repetitive structures and closed-loop sequences.

The model solution processes the set of calibrated images. The feature extraction is processed using the SIFT algorithm and relative camera poses are obtained using the standard five-point solver as described in [20] within the RANSAC loop in combination with bundle adjustments.

Relative camera poses are then decomposed into relative translations and relative rotations. Large errors among the obtained relative rotations are filtered with use of maximal spanning tree of relative camera rotations between image pairs. As the weight of camera edges the number of common features in the images are used.

After the spanning tree is created the excluded edges are added to the cycle and the check is performed to ensure, that the newly created cycle fulfills the consistency condition based on the angular difference metric commonly used on group of 3D-Rotations.

Each edge that does not fulfill the consistency condition is expected to be outlier. If the initial spanning tree contains the outlier rotation, it is possible that the correct rotations were identified as outliers. For this reason the additional search heuristics are applied to the rotational consistency algorithm and each outlier edge is added to the spanning tree in order to obtain better result.

Remaining relative rotations are processed in form of the quaternions and the absolute pose is obtained by minimizing the sum of squared angular errors.

Spatial reconstruction itself is ensured by using SOCP algorithm and bundle adjustments are processed to improve the final 3D reconstruction.

3.2 Camera calibration

Camera intrinsic parameters are obtained using the provided OpenCV library's function. The set of calibration images is loaded and the data from standard chessboard calibration pattern is obtained. Once the camera intrinsic parameters are obtained, the reconstruction images are undistorted for further processing.

3.3 Feature extraction and matching

Choice of optimal feature detector depends on the awaited attributes of input image set. It is expected that the images were obtained with camera with same intrinsic parameters and are calibrated. It can also be expected that pictures obtained are in same physical

orientation (most probably landscape) and with only the minimal angle differences in upward and forward axis.

Several methods for feature extraction were tested including the SIFT and FAST feature extractors and sparse and dense variants of Optical Flow patterns. Although the methods were providing good results, the highest number of correctly matched feature was obtained using the combination of FAST feature extractor, ORB descriptor and FLANN based matcher. The example results of different types of feature matching methods are listed in appendix F.

The appropriate filtration of matched features was essential for further processing. The median distances μ_x and μ_y for x and y axis of matched features were determined. The matches with the axial distances lower than $0.3 \cdot \mu$ and higher than $1.7 \cdot \mu$ were filtered out.

3.4 Relative pose calculation

Two approaches for obtaining the relative camera positions were tested - the five-point algorithm described in chapter 2.4.5 and the eight-point algorithm presented in chapter 2.4.4.

Five-point algorithm estimates the essential matrix $\mathbf{E}$ directly, while eight-point algorithm determines the fundamental matrix $\mathbf{F}$ first and estimates the essential matrix from the relationship 11 using the calibration matrices $\mathbf{K}$ and $\mathbf{K}'$. Five-point algorithm was able to estimate the essential matrix in most of the cases, however the RANSAC version of the eight-point algorithm provided more stable results.

3.5 Absolute pose calculation

Model solution absolute pose calculation is based on the minimization of error amongst the rotations. The relative orientations between cameras are represented as an undirected weighted graph, where the nodes represent the cameras, the edges are rotations and the number of matched features is used as the weight metric.

The evaluation of rotational consistency is based on the angular distance, the usual metric on 3D-rotations (often referred as SO(3)) that is defined as

$$d(\mathbf{R}, \mathbf{S}) = \underset{x \epsilon \mathbb{R}^3 \setminus \{0\}}{max} < (\mathbf{R_x}, \mathbf{S_x}) \,, \tag{18}$$

where $< (x, y)$ denotes the angle between the vectors x and y takes the value in the range of $[0, \pi]$. In case of relative rotations, where first camera is represented by unit matrix I and the cameras are rotated in one axis only with the angle α from range $[-\pi, \pi]$, then the $d(\mathbf{R}, \mathbf{I}) = |\alpha|$.

In the case of exactly known relative orientation $\mathbf{R_{ij}}$ between the cameras $\mathbf{R_i}$ and $\mathbf{R_j}$ the equation for all rotations of the graph would be

$$\mathbf{R_i} = \mathbf{R_{ij}} \mathbf{R_j} \,.$$

Since the results of matching process could not be considered as exact, there is a need to deal with low-level noise as well as completely inconsistent rotations – the outlier edges in the camera graph.

3.5.1 Model solution – handling the outlier rotations

In model solution the maximum-weight spanning tree $\mathbf{T}$ of orientation graph is created at first. Addition of the rotation e from set $\mathbf{E} \setminus \mathbf{T}$ then forms the cycle C. The absolute rotations amongst the cycle C are computed – the rotation $\mathbf{R_C}$ denotes the overall rotation of the whole cycle. In the ideal case the value of $\mathbf{R_C}$ should be $\mathbf{R_C} = \mathbf{I}$, since all the rotations in the cycle connects all the cameras in the graph. The angular distance is measured and checked if fulfills the condition

$$d(\mathbf{R_C}, \mathbf{I}) < \sqrt{|C|}\epsilon\,, \tag{19}$$

where $|C|$ denotes the size of the cycle and epsilon is the average error amongst the graph obtained that can be expressed as

$$\epsilon = d(\mathbf{R_{i1i2}}\mathbf{R_{i2i3}}.....\mathbf{R_{ini1}}, \mathbf{I})/n\,. \tag{20}$$

If C fulfills the condition (19), the edge e is then considered to be an inlier rotation and is putted into the set of inlier edges $\mathbf{E_C}$, otherwise it is excluded and considered as outlier. When all the edges that did not belong into the initial spanning tree are verified, the absolute rotations are estimated using the minimization described in following subsection.

In case that the initial edge forming the spanning tree was an outlier, the whole spanning tree would contain only the outlier rotations. For this reason, the spanning tree is created and processed also for all of the remaining outlier edges.

3.5.2 Model solution – handling low-level noise

The rotations in the inlier edge set are represented as unit quaternions and the camera positions are estimated using the approximate sum of squared angular errors. The computation using the quaternion representation may cause the ambiguity problem, however previously performed elimination of the outlier rotations provides the estimates $\bar{q_i}$ for the camera orientations.

The low-level noise handling process used in model solution is following:

- for each image pair (i, j) of the inlier rotation set,
  - represent the rotation $\tilde{\mathbf{R}}_{\mathbf{ij}}$ between (i, j) with quaternion $\tilde{q}_{ij}$,
  - compute $\tilde{\mathbf{Q}}_{\mathbf{ij}}$ as the matrix representation of $\tilde{q}_{ij}$,
  - if $\left|\bar{q_i} - \tilde{\mathbf{Q}}_{\mathbf{ij}}\bar{q_j}\right| > \left|\bar{q_i} + \tilde{\mathbf{Q}}_{\mathbf{ij}}\bar{q_j}\right|$ set $\tilde{\mathbf{Q}}_{\mathbf{ij}} = -\tilde{\mathbf{Q}}_{\mathbf{ij}}$.

3.5.3 Proposed solution

Proposed solution uses the same method for handling the outlier rotations, but solution for handling the low-level noise is more straightforward using the computation for all possible combinations. Outlier rotations are also handled using the spanning tree followed by creating cycles using the leftover edges and comparing them to the rotational error tolerance. The low-level noise is handled simply by searching the graph for cycle with the lowest rotational error. This is done by generating all possible edge combination and checking the error threshold.

After finding the best solution for the initial maximum spanning tree, the outlier edges are processed the same way as the model solution. Once the combination with the lowest rotational error is found, the absolute camera positions are iteratively computed.

3.6 Projection reconstruction

For estimation of the point's spatial positions the proposed solution uses the Linear-LS. Although this algorithm is not the best solution for projective reconstruction, it is quite fast and provides satisfactory results. Triangulated point positions are stored in a point cloud structure along with its color information.

3.7 Surface triangulation

The point cloud is filtered out for outliers – the places with low density of points – and then the surface is triangulated using the greedy algorithm described in chapter 2.8.

4 Implementation

This chapter describes the implementation detail of proposed solution. First part presents the technologies and frameworks used, later parts describe the implementation details including the structures and algorithms proposed in chapter 3.

4.1 Used tools and frameworks

The application was programmed in Microsoft Visual C++ using platform toolset v110. The OpenCV and Point Cloud Library libraries were used.

4.1.1 OpenCV

OpenCV is an open source library providing the implementation of most popular and commonly used algorithms in computer vision applications, including the algorithms for face recognition, image manipulation or camera tracking.

Important features implemented in the library are camera calibration, feature extraction and correspondence matching, which will be used in this work.

4.1.2 Point Cloud Library

Point Cloud Library is an open library that provides wide range of tools focused mainly on work with point clouds – multidimensional point structures. Aside the space coordinates, each point of the cloud can contain more information, such as color, temperature or directional vector. This makes the point cloud structure very flexible and usable in a wide range of applications.

Thanks to native support of the OpenNI framework, the Point Cloud Library can process the input from several compliant 3D sensors and create the point cloud in real time. One of the Point Cloud Library's key features, which is important for this thesis is the algorithm for fast triangulation of unordered point clouds. This allows to calculate the surface of the input point cloud data and generate the 3D mesh for further use. Aside of that, Point Cloud Library also provide the algorithms for filtering the noise points in point clouds using the statistical analysis methods, normal estimations, surface smoothing, correspondence grouping and many others.

4.2 Application architecture overview

The application illustrating the proposed solution is divided into several classes in order to group the functionality into following steps.

4.2.1 Calibration

The Calibration class defines the methods used for extracting the calibration camera parameters and application of image corrections on input images. **CalibCamera** method accepts the input in the form of calibration images paths, extracts the camera intrinsic

parameters and returns the calibration matrix and the matrix of distortion coefficients using the referenced output matrices. The **UndistortImage** method encapsulates the undistortion of the image using the previously acquired calibration matrix and matrix of distortion coefficients.

4.2.2 ImageCouple

The ImageCouple structure keeps information about an image pair. The images are referenced using the pointers. The keypoints for both images are stored along with the information about their pairing. Except the above mentioned, the ImageCouple structure also keeps the information about the fundamental and essential matrices, relative camera matrix and the relative rotation and translation matrices.

4.2.3 ImageCollection

The ImageCollection class is the basic structure used in whole application. It stores the original image, calibrated image data, features found in each image, absolute camera positions and the information about image couples.

ImageCollection is created using the plain constructor without any parameters. Once initiated, the images are loaded using the **LoadImages** method that accepts the collection of the paths leading to particular images as an input. The camera calibration is processed by the **CalibrateCamera** method processing the input of the collection of paths leading to the calibration images. **CalibrateImages** method undistorts the set of input images. The undistorted image set is stored among with its greyscale variant later used for feature extraction.

InitImageMatches method covers the feature extraction and matching. Features from each image are extracted and described and then the image pairs are searched for correspondences. The information about corresponding features are stored in the ImageCouple structure.

FindFundamentalMatrices method covers the process of estimation of the fundamental matrices for each ImageCouple. Once the fundamental matrices are found, the essential matrices are estimated using the **FindEssentialMatrices** method and the camera relative rotation and translation is estimated for each ImageCouple using the **RecoverPoses** method.

4.2.4 FeaturesMatcher

FeaturesMatcher class provides the methods for features extraction for a single image and features matching for an image couple.

The features are extracted using the **ExtractFeatures** method. The reference to the image data is passed as an input, and features are extracted and described. The positions and descriptors of the points are returned using the referenced output vectors. **FindImageCoupleMatches** method compares the referenced feature descriptors for im-

ages in image pair and stores the information about matched features in the referenced ImageCouple structure.

4.2.5 CameraPose

CameraPose class encapsulates the computation of the relative rotation and translation of image pair using the knowledge of the essential matrix and image correspondences.

The **decomposeEssentialMat** decomposes the essential matrix itself. Since the decomposition output is not unambiguous, the calling of the method is encapsulated by the **recoverPose** method that verifies the possible solution using the cheirality check. Both methods implementations are modified implementations of the methods used in the future release of OpenCV library.

4.2.6 Edge

Edge structure is defined as a part of Graph class and serves as a simple container that keeps the information about the Graph nodes it connects. Edge structure basically mirrors the ImageCouple structure, however it is adapted for effortless use in the Graph class.

The Edge structure keeps the string identifier in the form "**cam1id#cam2id**" where **cam1id** and **cam2id** are the string representation of pointers leading to the calibrated images stored in ImageCollection object. Image ids are also stored separately.

Each Edge also stores the relative rotation and relative translation matrices along with the number of correspondences between the images. The number of correspondences is used as a weight of the Edge. The operators **<** and **>** were overridden to allow comparing of the Edges based on this metric.

4.2.7 Graph

Graph class contains the methods for working with the camera graphs used in application. As was mentioned above, the Graph class also contains the definition for Edge structure.

The Graph's constructor uses the Edge set reference as an input. The nodes are simply referenced using the string representation of pointer to specific image file stored in ImageCollection's calibrated images set. Graph class also keeps the absolute rotational and position matrices for each node.

InitAdjacency method constructs the internal array of integer list used to express the relationships between the particular nodes based on the set of Edge indices provided as input. The each node's integer list represents the indices of connected nodes.

Once the adjacency of the provided Edge combination is initiated graph can be tested for the presence of the cycles using the **IsCombinationCyclic** method. The absolute camera rotations and positions for the adjacency combination are computed using the **ComputeAbsoluteCamera** method.

4.2.8 Geometry

Geometry class contains the methods for processing the image triangulation itself. References to the ImageCollection, the Edge set, inlier Edge indices and absolute rotation and translation matrices are passed as input. The inputs are processed in the **CreatePointCloud** method that generates the PCL's PointCloud structure with the reconstructed spatial point information.

The point cloud can be further processed using the methods **FilterPointCloud** that filters out the points in the low density areas, and TriangulateSurface which generate the mesh. The point cloud can be retrieved using the **GetPointCloud** method and the final mesh can be accessed using the **GetMesh**.

4.2.9 SRUtils

SRUtils is a helper class containing the methods used across the whole application. **printMat** method provides the readable string output of the cv::Mat's values. **displayImage** serves to display a single image in a reasonably sized window and wait for the user's action. **displayImages** is used to display a collection of images using the displayImage method.

4.3 Overall workflow

The application's input are two sets of images. First set of images contains the chessboard calibration pattern in shot under different angles. These images are used for the calibration – estimation of intrinsic camera parameters. The second set of images contains the images for the spatial reconstruction. It is required that the calibration images and the reconstruction images are acquired with the same camera settings.

Algorithm 1 Overal workflow

Workflow:

- Estimation of calibration data
  - Loading the input set of calibration images
  - Determination of calibration matrix and distortion coefficients
- Processing the input set of reconstruction images
  - Loading the input set of reconstruction images
  - Undistorting the input set of reconstruction images
  - Extracting the features for each image from undistorted reconstruction set
  - Matching the features across the all possible combination of image pairs
  - Estimation of each image pair's fundamental matrix Estimation of each image pair's essential matrix
  - Recovering the relative poses of cameras in each image pair
- Absolute poses estimation amongst the camera cycle
  - Creating the set of graph edges containing the information about connected cameras
  - Sorting the edge set descending by weight (number of matched features)
  - Computing the total angle error amongst the camera cycle
  - Computing the maximum spanning tree using the edge set
  - Determination of the inliers and outliers edge sets
  - Estimation of the best camera cycle combination
  - Additional search heuristics applied on outlier edges
- Spatial reconstruction
  - Final estimation of absolute camera poses
  - Iterative triangulation and creation of point cloud
- Point cloud visualization and storing
  - Point cloud surface triangulation
  - Displaying the graphical output

4.4 Estimation of calibration data

Calibration images paths are obtained and passed to the Calibration's module CalibCamera method. Each image is loaded as cvMat, checked for chessboard using cv::findChessboardCorners method pattern, the positions of found corners is particularized using cv::cornerSubPix method. The determination of calibration matrix and distortion coefficients is done using cv::calibrateCamera method. The obtained calibration matrix and distortion coefficients are stored in ImageCollection.

Algorithm 2 Estimation of calibration data

- For each calibration image
  - Load image into memory
  - Search for the calibration pattern
  - Extracting more precise position information of found corners and storing it for further processing
- Obtaining and storing calibration matrix and distortion coefficients

4.5 Processing the input set of reconstruction images

Reconstruction images are loaded and stored in ImageCollection, simultaneously the images are undistorted using the cv::initUndistortRectifyMap and cv::remap using the previously obtained distortion coefficients. Undistorted images are stored in the ImageCollection as well as their greyscale version. Features are extracted and described from each undistorted greyscale image using the OpenCV's SIFT feature extractor and descriptor. Features locations and descriptions are stored in ImageCollection.

Algorithm 3 Features extraction

- For each reconstruction image
  - Original image is loaded into memory and stored in ImageCollection
  - Undistorted copy of image is created and stored in ImageCollection
  - Greyscale version of undistorted image is created and stored in ImageCollection
  - Features are extracted from greyscale version of undistorted image and stored in ImageCollection

When all the input images are processed, the ImageCouple structure is created for each image pair combination. Using the previously extracted features, each

ImageCouples are matched using the OpenCV's Flann matcher. Found correspondences are filtered using minimal and maximal translation distance.

The medians for distances in x-axis and y-axis were determined and the matches with axial distances lower than minimal treshold value or higher than maximal threshold value were filtered out. The minimal threshold value was set to 0.3 multiple of the median, while the maximal threshold was set to 1.7 multiple of the median.

Using the filtered image correspondences the fundamental matrix is estimated via the cv::findFundamentalMat method using the RANSAC version of eight-point algorithm. Essential matrix is then determined from the equation (11).

Algorithm 4 Features matching and estimation of relative positions

- For each image pair
  - ImageCouple structure is created
  - Image features are matched, filtered and stored in ImageCouple
  - Fundamental matrix is estimated using the matched features stored in ImageCouple
  - Essential matrix is estimated stored in ImageCouple
  - Relative camera rotation and translation are determined stored in ImageCouple
  - ImageCouple is stored in ImageCollection

4.6 Absolute poses estimation

Once all the image pairs are processed and relative rotations and positions are estimated, the process of absolute poses estimation begins. The set of Edge structures is created, each Edge stores the information about relative rotation and number of found image correspondences for one image pair.

The total angle error and error tolerance are then computed according to the equations (19) and (20) from section 3.5. The set of Edges is ordered in descending order using the number of correspondences as the weight and the maximum spanning tree for the edge set is created.

Algorithm 5 Absolute camera positions - preparation

- For each ImageCouple
  - Edge structure is created storing the information about the connected images, relative camera rotation and number of found correspondences
- The total angle error and error tolerance are computed
- Set of edges is ordered in descendant order by the number of correspondences
- The maximum spanning tree is created

Once the maximum spanning tree is created, the rest of the edges are added one by one to the spanning tree creating a cyclic graph. The rotational error for the cycle is computed and compared to the error tolerance. If the error in cycle is lower than threshold, the edge is listed inlier and is added to the final camera relationship graph.

When the camera relationship graph is created, all the possible cycles connecting all points in the graph are checked and overall rotational error is computed. The combination with the lowest rotational error is selected as a best solution.

Algorithm 6 Best camera cycle estimation

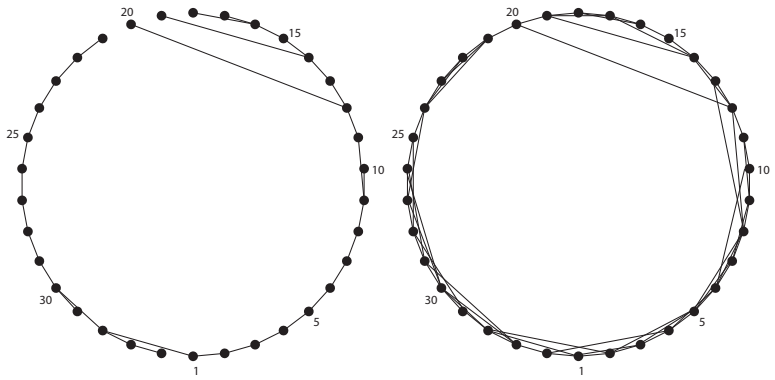
1. For each edge that doesn't belong to the spanning tree
   (a) The cycle is formed using the spanning tree and the edge
   (b) If the cycle's angle distance is lower than error tolerance, the edge is classified as the inlier
2. The graph is created using the spanning tree and inlier edges
3. Angle distance for every combination of possible ways through each point of graph is estimated
4. The combination with the lowest angle distance is picked as a right solution
5. Additional search heuristics are applied

Once the process is done for the maximum spanning tree, the additional search heuristics are applied. For each outlier edge a new spanning tree is created including the current outlier edge. Then the steps 1-4 from algorithm (6) are repeated for a newly formed spanning tree in order to find a solution with lower overall rotational error.

Algorithm 7 Additional search heuristics

- For each outlier edge
  - New spanning tree is created starting with current outlier edge
  - Estimating the lowest overall rotational error - steps 1-4 from algorithm (6)

Figure 2 shows the initial spanning tree of the tank reconstruction set (appendix A) and the camera circle after the selection of best possible combination of cameras. The cameras have been place on a circle using the prior knowledge of the true geometry in this illustration.



(a) Initial spanning tree (b) Filtered camera circle

Figure 2: Filtered camera circle for tank reconstruction set

When the combination of edges with the lowest overall rotational error is found, it is used to compute the absolute internal camera matrices using the knowledge of relationships between linked cameras its and relative rotations and translations.

4.7 Spatial reconstruction

Using knowledge of absolute internal camera matrices and image correspondences between paired cameras, the spatial reconstruction can be processed. The Point Cloud Library's PointCloud is used to store information about individual points including the position and color, which is retrieved as average value of corresponding image pair's points color.

The data about related image pairs are loaded from ImageCollection using the previously processed Edge set. For each Edge in the set the image data and correspondences are obtained. This data are further processed by the TriangulatePoints method - the more detailed process of the method is described in algorithm (9).

Once the PointCloud is created, the PCL's StatisticalOutlierRemoval is used to filter out the noise points and the surface is triangulated using the PCL's GreedyProjectionTriangulation. The resulted mesh is stored using PCL's PolygonMesh.

Algorithm 8 Spatial reconstruction

- Define PCL's PointCloud for all points
- For each Edge in inlier edge set
  - Retrieve the original images
  - Retrieve image correspondences
  - Triangulate the spatial positions for image pair using TriangulatePoints method and add the points into PointCloud
- Filter out the outlier points of the PointCloud
- Triangulate the mesh surface of the PointCloud and store it as PolygonMesh
- Export the PolygonMesh
- Visualize the PolygonMesh

At last, the PolygonMesh is exported and saved to the hard drive using the PCL's savePLYFileASCII method and visualized using PCL's CloudViewer.

4.7.1 TriangulatePoints method details

The presented TriangulatePoints method is based on the implementation of Harltey and Zisserman's Linear-LS triangulation [10] presented in [6], modified to work with PCL's PointCloud.

Algorithm 9 Point triangulation details

Input

- References to keypoint set pt_set1 and corresponding keypoint set pt_set2
- Reference to calibration matrix K
- References to camera internal matrices P1 and P2
- References to images I1 and I2
- Reference to output PointCloud

Output

- Mean of reprojection error

Algorithm

- Resetting the output PointCloud
- Initializing the internal variables
- For each correspondence
  - Conversion of referenced image points into normalized homogenous coordinates
  - Construction of equation matrices
  - Solving the matrices using SVD decomposition and obtaining the point's spatial coordinates
  - Estimation of point's color
  - Storing the point's information in output PointCloud
- Reprojection error mean computation

5 Input requirements

Both input images are subjected to several restrictions. Both image sets are supposed to be acquired with same camera using the constant settings, especially the focal length and resolution. Exposition time is irrelevant.

5.1 Calibration image set

The calibration algorithm expects the calibration set to contain the chessboard calibration pattern with 6 columns and 5 rows and square's edge size about 3 cm. The calibration set should contain about 10 images, each should be nearly filled with the calibration pattern and each should be acquired from different angle. Larger amount of calibration images enhances the precision of calibration.

5.2 Reconstruction set

Camera settings should be constant during the acquisition of reconstruction images, focal length must be same as in the case of calibration images. It is recommended to use images with short baseline, since image pairs with short baseline contains significantly higher amount of correspondences. The camera's sensor plane should be perpendicular to the ground. It is possible to process the images taken with camera pointed up, however due to the unknown angle the spatial reconstruction will be distorted.

6 Results

This chapter compares the performance of proposed solution with already existing solution on selected input sets of images. The proposed solution is compared to Autodesk 123D Catch software.

6.1 Reconstruction sets

Three sets of input images were acquired and processed using reference solution and the proposed application. Tank and church reconstruction sets (appendix A and appendix B) were acquired according to the recommendations from chapter 5.2, the wooden figure reconstruction set (appendix C) was acquired by rotationg the figure itself on the static background.

The camera configurations used for obtaining reconstuction sets are listed in tables 1 (tank and church) and 2 (wooden figure). All the reconstructions were processed on the configuration listed in the table 3.

6.1.1 Tank reconstruction set

The first reconstruction set contains 34 images of the army tank. The images were taken in the strict order around the object. Images are listed in appendix A.

Processing the tank reconstruction set, large number of features was found and correctly matched. However, the matched features contained lots of features of the ground, that negatively affected the result.

The final result of tank set reconstruction is displayed in figure 3. Large amount of the features was obtained and triangulated. The shape of the displayed point cloud corresponds with the expected shape of the original object and its surrounding. The reconstruction of the tank set took 4 hours and 43 minutes.

6.1.2 Church reconstruction set

Church reconstruction set contains 34 images of the church. 22 images were taken around the church itself, remaining 12 images display details of the entrance. The images are listed in appendix B.

The set did not contained enough images for complete reconstruction. Also the camera pose estimation was complicated by the repeating patterns in the structure of the church. The 12 images of the entrance were not estimated correctly and only generated noise in the resuting point cloud.

The final church point cloud was reconstructed using only the right and back side of the church (images 29-34). The density of the point cloud is quite sparse, however the basic shapes are recognizable. The result is shown in figure 4. The reconstruction of the church set took 56 minutes.

Figure 3: Reconstruction of the tank set

6.1.3 Wooden figure reconstruction set

Wooden figure reconstruction set was acquired using the static camera and by rotating the object. The reconstruction set contains 10 images, that are listed in appendix C.

Due to the method of acquiring the reconstruction set and low contrast of the wooden figure itself, the number of matched features was very low, which resulted into incorrect estimation of the camera positions.

The reconstruction itself failed and the nonsencial point cloud was created. The result is shown in figure 5. The reconstruction took 2 hour and 5 minutes.

6.2 Comparison with existing solution

All three reconstruction sets were also processed using the Autodesk's 123D Catch software. The reconstruction itself is performed on the server side, therefore it is impossible to determine the exact reconstruction time. The reconstruction time is also influenced by the number of other currently processed reconstructions. However, each set was processed within two hours after the uploading of the images.

6.2.1 Tank reconstruction set

AutoDesk's 123D Catch handled the tank reconstruction well. The resulted mesh contained large numer of vertices. According to the density of vertices it is probable that the surface tessalation was applied on the final mesh. The result of the tank reconstruction set can be seen in figure 6.

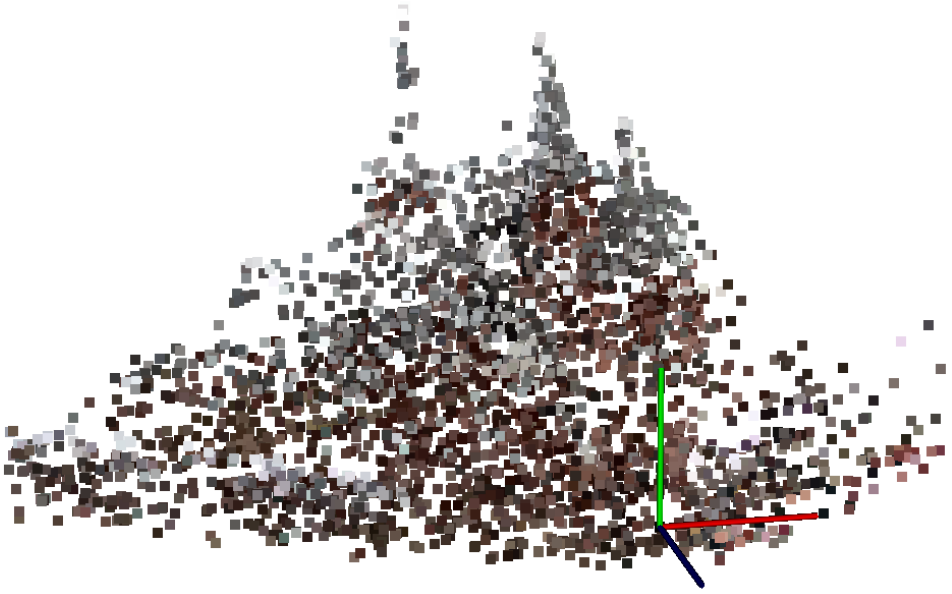


Figure 4: Reconstruction of the church set

Figure 5: Reconstruction of the wooden figure set

6.2.2 Church reconstruction set

The church reconstruction set was not reconstructed correctly. Since the images contained repetetive structure componens, the 123D Catch was not able to detect the positions of the cameras correctly. Only the right and back side were reconstructed and the final mesh contained a large amount of incorrectly triangulated vertices and the mismapped textures.

The slight improvement of the reconstruction was achieved by using only the images of the right and back side of the church (images 29-34). Figures 7 and 8 shows the results of the church reconstruction set.

6.2.3 Figure reconstruction set

Autodesk 123D Catch also failed processing the figure reconstruction set. No model was created.

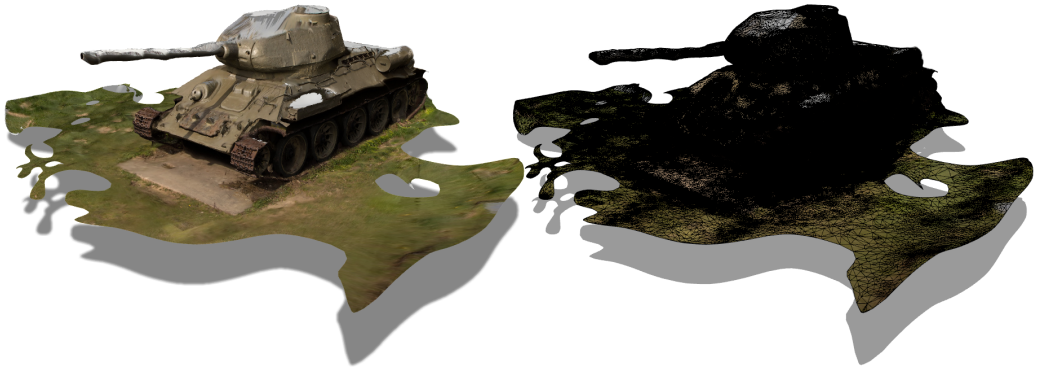


Figure 6: Autodesk 123D Catch reconstruction of tank set

Figure 7: Autodesk 123D Catch reconstruction of church set

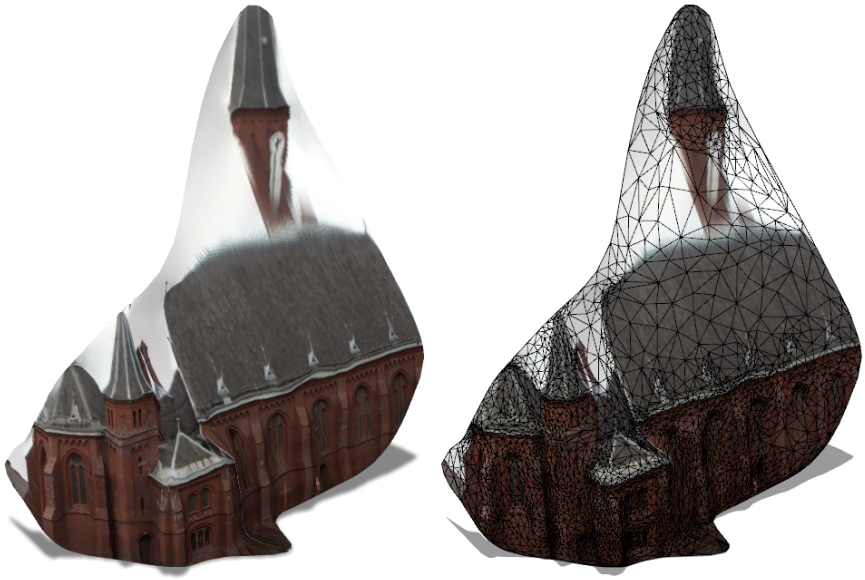


Figure 8: Autodesk 123D Catch reconstruction of limited church set

6.3 Evaluation of the results

As was expected the quality of the 123D Catch's results is high. According to the topology of the generated models, it can be presumed that the software uses additional surface tessalation to achieve higher density of the final mesh. This is especially noticable on the models generated from the church reconstruction set (figures 7 and 8), where it is highly improbable that the incorrectly reconstructed sky would contain such a high number of matched features.

The pointclouds generated by the demonstrational application presented in this thesis have sufficient density and the topology of points visually corresponds with the topology of the real scene objects. It can be expected that filtering out the areas with low point density and triangulating the pointcloud's surface, the topology of created mesh could achieve the desired precision and after tessalation it would mach the quality of the meshes generated by Autodesk 123D Catch.

7 Conclusion

The thesis takes focus on common approaches of spatial reconstruction from calibrated image set. The algorithms solving the particular steps of the reconstruction process workflow were described, compared and tested. The demonstrational application was created and compared to the existing commercial solution.

The introduction of the thesis briefly describes the possibilities of the spatial reconstruction using several different approaches and introduces the main ideas behind the structure from motion pattern. Also the current state of the art solutions are introduced.

Following part of the thesis the basic principles of the structure from motion are presented along with several approaches of solving the particular steps of the reconstruction workflow. Commonly used algorithms and patterns are explained and compared.

The proposed demonstrational application is based on the presented model solution and the details of its implementation are described. Unfortunately due to the unexpected complexity of assignment the surface triangulation and texturing have not been implemented.

The results of the demonstrational application are presented and compared to the existing commercial solution. Although the demonstational application creates only the point cloud, the shapes are identifiable and it is expected that after filtering the areas with low density of points and the surface triangulation, the topology of the final mesh would correspond with the real structure of reconstructed object.

The results of the reconstuction itself are satisfactory, however there are numerous ways in order to improve the application's effectivity and accuracy. The main objective of the future developent of the application should concern about the finishing the application's output - the pointcloud should be filtered and triangulated to produce the mesh. Also the texture should be prepared and mapped on the final mesh.

Further development could then be focused on improving the reconstruction speed, for example by using the OpenCL or CUDA technologies. The usage of better spatial triangulation algorithm could improve the stability of reconstruction and allow to process the reconstruction sets that currently provide incorrect results.

8 References

[1] Fast algorithm for corner detection. [Online]. Available: http://docs.opencv.org/trunk/doc/py_tutorials/py_feature2d/py_fast/py_fast.html(2014/02/09)

[2] Introduction to sift (scale-invariant feature transform). [Online]. Available: http://docs.opencv.org/trunk/doc/py_tutorials/py_feature2d/py_sift_intro/py_sift_intro.html(2014/02/08)

[3] Introduction to surf (speeded-up robust features). [Online]. Available: http://docs.opencv.org/trunk/doc/py_tutorials/py_feature2d/py_surf_intro/py_surf_intro.html(2014/02/08)

[4] Optical flow. [Online]. Available: http://docs.opencv.org/trunk/doc/py_tutorials/py_video/py_lucas_kanade/py_lucas_kanade.html(2014/04/05)

[5] (2011, January) Comparison of the opencv's feature detection algorithms. [Online]. Available: http://computer-vision-talks.com/articles/2011-01-04-comparison-of-the-opencv-feature-detection-algorithms/(2014/02/08)

[6] D. L. Baggio and A. Zisserman, *Mastering OpenCV with practical computer vision projects*, 2nd ed. Birmingham, UK: Packt Pub., 2012.

[7] H. Bay, T. Tuytelaars, and L. V. Gool, "Speeded-up robust features (surf)," *Computer Vision and Image Understanding*, vol. 110, pp. 346–359, 2008.

[8] J. Dolz. (2011, May) Image distortion. [Online]. Available: http://josedolz.jimdo.com/computer-vision-tutorials/image-distortion/(2014/01/15)

[9] O. Enqvist, F. Kahl, and C. Olsson, "Non-sequential structure from motion," in *OMNIVIS*, 2011.

[10] R. Hartley and P. Sturm, "Triangulation," *Comput. Vis. Image Underst.*, vol. 68, no. 2, pp. 146–157, Nov. 1997. [Online]. Available: http://dx.doi.org/10.1006/cviu.1997.0547

[11] R. Hartley and A. Zisserman, *Multiple view geometry in computer vision*. Cambridge University Press, 2003.

[12] H. Li and R. Hartley, "Five-point motion estimation made easy," in *ICPR '06 Proceedings of the 18th International Conference on Pattern Recognition*, 2006.

[13] H. C. Longuet-Higgins, *Readings in Computer Vision: Issues, Problems, Principles, and Paradigms*. Morgan Kaufmann Publishers Inc., 1987, ch. A Computer Algorithm for Reconstructing a Scene from Two Projections, pp. 61–62.

[14] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, pp. 91–110, 2004.

[15] B. D. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision," in *Proceedings of Imaging Understanding Workshop*, 1981.

[16] Z. C. Marton, R. B. Rusu, and M. Beetz, "On fast surface reconstruction methods for large and noisy point clouds," in *Proceedings of the 2009 IEEE International Conference on Robotics and Automation*, ser. ICRA'09. Piscataway, NJ, USA: IEEE Press, 2009, pp. 2829–2834. [Online]. Available: http://dl.acm.org/citation.cfm?id=1703775.1703907

[17] J. J. More, "Levenberg–marquardt algorithm: Implementation and theory," in *Conference on numerical analysis, Dundee, UK*, 1977.

[18] M. Muja and D. G. Lowe, "Fast approximate nearest neighbors with automatic algorithm configuration," in *In VISAPP International Conference on Computer Vision Theory and Applications*, 2009, pp. 331–340.

[19] K. H. Nghia. (2012, September) Five point algorithm for essential matrix, 1 year later. [Online]. Available: http://nghiaho.com/?p=1675(2014/02/02)

[20] D. Nistér, "An efficient solution to the five-point relative pose problem," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, 2004.

[21] P. M. Pancha, S. R. Panchal, and S. K. Shah, "A comparison of sift and surf," *International Journal of Innovative Research in Computer and Communication Engineering*, vol. 1, 2013.

[22] E. Rosten and T. Drummond, "Machine learning for high-speed corner detection," in *Computer Vision - ECCV 2006*, 2006.

[23] U. Sinha. (2010, July) Two major physical defects in cameras. [Online]. Available: http://www.aishack.in/2010/07/two-major-physical-defects-in-cameras/ (2014/01/15)

[24] N. Snavely, S. M. Seitz, and R. Szeliski, "Modeling the world from internet photo collections," *International Journal of Computer Vision (IJCV)*, 2007.

[25] Z. Zhang, "A flexible new technique for camera calibration," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2000.

[26] Z. Zhang, *Emerging topics in computer vision*. Prentice Hall PTR, 2004, ch. Camera Calibration, pp. 4–43.

A Tank reconstruction set


1


2


3


4


5


6


7


8


9


10


11


12


13


14


15


16


17


18


19


20


21


22


23


24


25


26


27


28


29


30


31


32


33


34

B Church reconstruction set


1


2


3


4


5


6


7


8


9


10


11


12


13


14


15


16


17


18


19


20


21


22


23


24


25


26


27


28


29


30


31


32


33


34

C Wooden figure reconstruction set


1


2


3


4


5


6


7


8


9


10

D Camera configurations

Camera	Canon 650D
Sensor format	APS-C; Crop factor 1.62x; Aspect ratio 3:2
Sensor size	22.3mm x 14.9mm
Lens	EF-S 18-135mm f/3.5-5.6 IS STM
Focal length	18mm (35mm format focal length equivalent: 30mm)
Mode	Aperture priority
Focusing	Automatic; Central focus point
Metering mode	Center-weighted average metering
Image quality	RAW
Resolution	5184 x 3456 pixels
ISO	100
White balance	Daylight

Table 1: Camera settings

Camera	Canon 650D
Sensor format	APS-C; Crop factor 1.62x; Aspect ratio 3:2
Sensor size	22.3mm x 14.9mm
Lens	EF-S 50mm f/1.8
Focal length	50mm (35mm format focal length equivalent: 80mm)
Mode	Aperture priority
Focusing	Manual
Metering mode	Spot metering
Image quality	RAW
Resolution	5184 x 3456 pixels
ISO	100
White balance	Shade

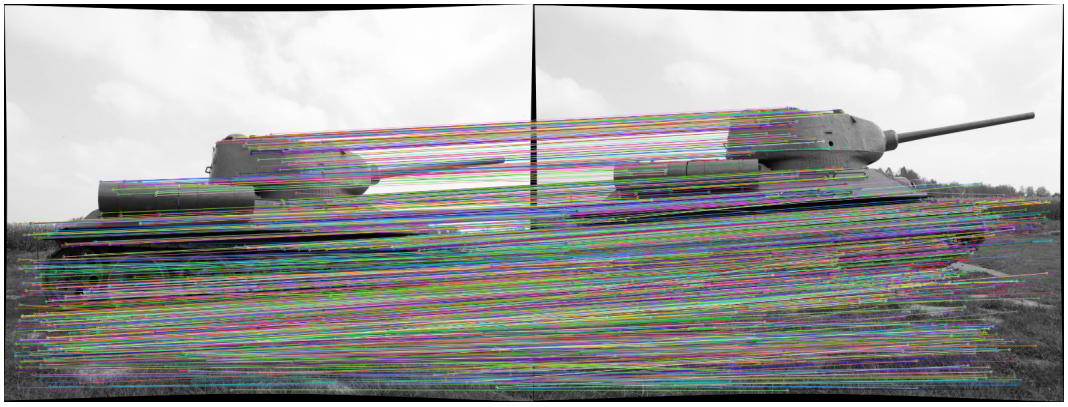
Table 2: Camera settings

E The testing computer configuration

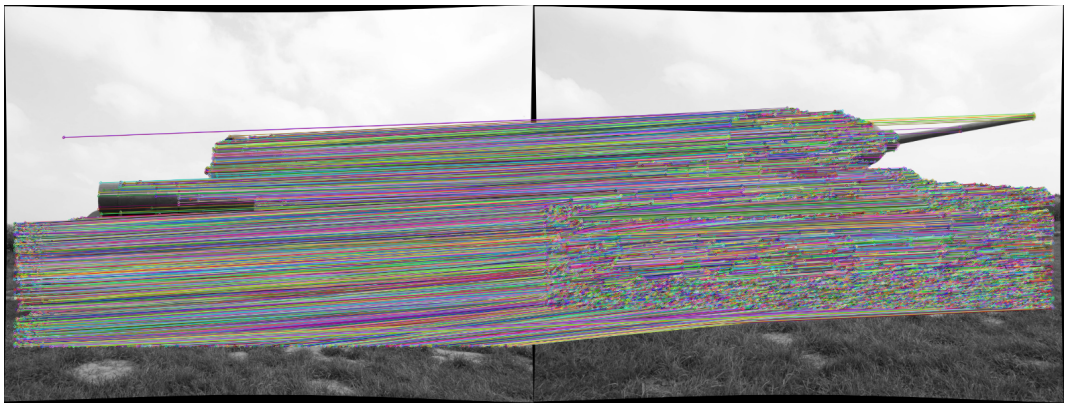
Manufacturer	Asus
Model	G55VW
CPU	Intel Core i7-3610QM; 4 cores; 2.3 GHz
RAM	SO-DIMM 24GB DDR3; 1600MHz; CL 11
Primary HDD	SanDisk SSD U100 128GB
GPU	NVIDIA GeForce GTX 660M
OS	Windows 8.1 Pro 64-bit

Table 3: Computer configuration

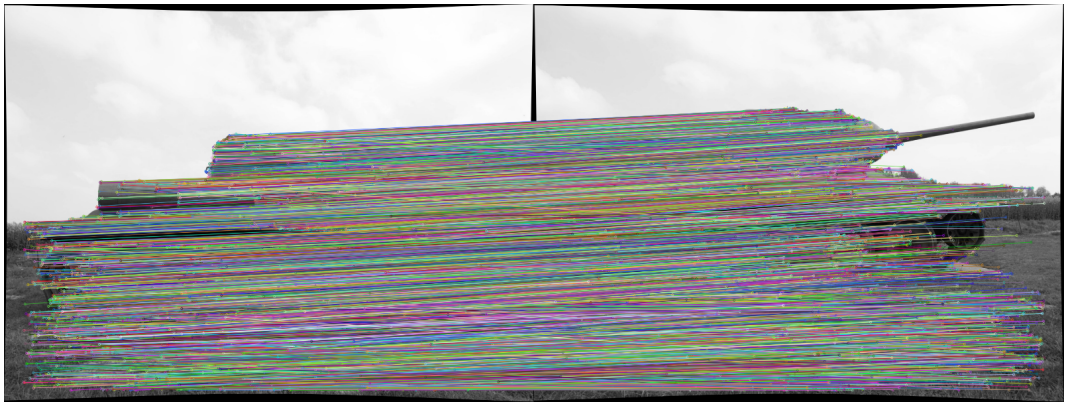
F Feature matching methods comparison



(a) Sparse optical flow



(b) Dense optical flow



(c) FAST feature extraction and FLANN matching

Figure 9: Comparison of feature matching methods

G Content of the attached DVD

Directory	Description
SCENE_RECONSTRUCTION	The folder containing the demonstrational application
SRC	Source codes of the demonstrational application
BIN	Binaries of the demonstrational application
LIB	The installation files of libraries necessary to compile and run the demonstrational application
INPUT	The folder containing the input data used in the thesis
CALIB_18	The images of calibration patterns for tank and church image sets
CALIB_50	The images of calibration patterns for wooden figure image set
REC_CHURCH	The church calibration set
REC_FIGURE	The wooden figure calibration set
REC_TANK	The the tank calibration set
Thesis.pdf	The PDF version of this thesis